連載（第１回）「論理合成（ロジック・シンセシス）」は「魔法の道具（マジック・アイテム）」にあらず

# 論理合成の基本技術（前編）

藤田昌宏

**「ロジック・シンセサイザ」，「シリコン・コンパイラ」といった言葉から受ける印象をうのみにして，論理合成ツールに対して過大な期待を抱くビギナのエンジニアは少なくない．おおざっぱに言うと，論理合成ツールはHDLで記述した仕様を論理式などに変換し，決められた手続きにのっとって，論理ゲート数や論理段数が減る方向へ回路を変形・置換しているにすぎない．上述のような誤解が広がりやすいのは，論理合成ツールをブラックボックスと見なして使用しているユーザが多いためだろう．本連載では論理合成ツールの内部処理に言及しながら，その特性を明らかにしていく．こうした原理・原則を理解していれば，論理合成ツールの実力と限界をだれでも見極められるようになる．**

**（編集部）**

本稿では，論理合成の基礎技術を理解し，論理合成ツールを上手く使いこなせる設計者になっていただくことを目標に，連載で解説していきます．さて，ここでいう「論理合成」とはどういうものを指すのでしょうか？ LSI設計では，論理回路からレイアウト（配置・配線）ツールを使ってマスク・パターンを作成します．そのため，あらかじめなんらかの方法で論理回路を作っておく必要があります．このとき，論理回路を直接，設計者が作成するのではなく，論理式や真理値表，状態遷移表などから「自動的」に論理回路を作成するツール（技術）が論理合成と呼ばれているものです．

本連載では，この論理合成技術について，以下の項目に沿って解説していきます．

1．論理合成の基本技術
2．ツールを用いた論理式からの合成体験
3．論理回路のタイミング最適化技術
4．ハードウェア記述言語からの合成
5．実際の論理合成ツールの機能と今後の動向

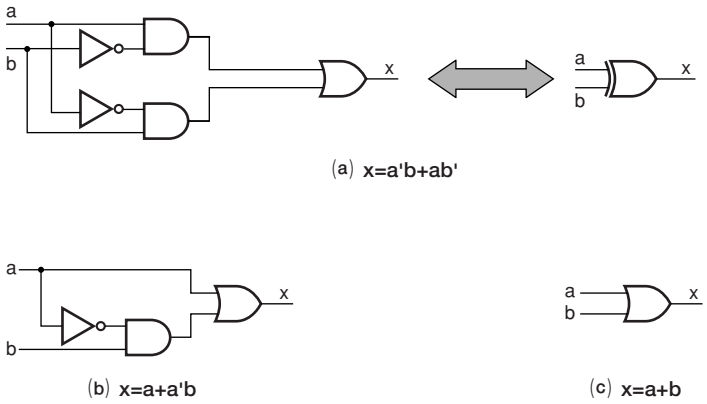
論理関数の処理などが出てくるので少しややこしい部分もありますが，最後まで読んでいただければ，論理合成技術の概要が理解でき，ツールの中でなにが行われているか，ツールからみてなにが簡単でなにがむずかしいか，論理合成ツールをどう取り扱えばより効率よく使いこなせるかなどを理解していただけると思います．

一般に，EDAツールがどんどん高機能化している一方で，高機能であるがゆえに使い方次第で結果に大きな差がでるようになってきています．設計品質をあげるため，また他の設計者よりもよりよい回路を生成するため，本連載が助けになると考えています．

多くの方が，米国Synopsys社のDesign Compilerに代表される市販の論理合成ツールを利用していると思います．論理合成ツールが使われ始めたのは1980年代ですから，ツールとしての実績は長く，動作も安定しており，論理合成ツールをブラックボックスとして考え，利用している人も多いでしょう．普通はそれで十分です．しかし，以下のような問題が生じることも少なくないのではないでしょうか．

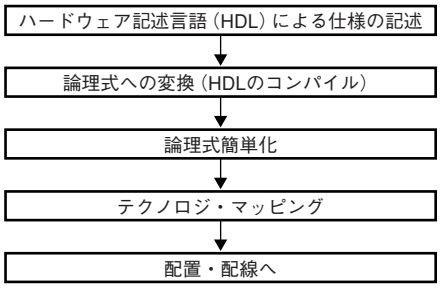
- いろいろパラメータを変えてやってみたが，どうしても回路が思ったほど小さくなってくれない．あるいは，タイミングが合わない．
- 既存の回路を一部修正して利用したいが，うまくいかない．どうしてよいかわからない．
- ハードウェア記述言語（HDL）の書き方で合成結果が大きく違い，トリッキだ．
- あまり大きな回路ではないのに，合成に長い時間がかかる．

こうした問題が生じる背景を理解していただくため，




〔図1〕論理関数と論理回路

(a)は論理関数 x=a'b+ab' を実現する2種類の回路を示している．基本的なゲートと言えるAND，OR，NOTのみから構成すると左の回路になるが，a'b+ab' は排他的論理和を表しているので，右のようにEXORゲート一つでも実現できる．(b)は論理関数x=a+a'b をそのまま回路に直したものであるが，じつはa+a'b=a+b なので，(c)のように回路を簡単化できる．




〔図2〕
**論理合成ツールによる論理回路設計の流れ**

論理回路設計は，まずハードウェア記述言語（HDL）で回路の動作を記述することからはじまる．それをHDLコンパイラによって論理関数へ変換する．次に論理式の簡単化を行い，最後にその論理関数を実際に使用する半導体技術用に用意されているゲートやセル（ライブラリと呼ばれる）を用いた回路に変換する．

本連載ではまず，論理合成の基本技術の説明から始めます．次に，その理解を確認するため，カリフォルニア大学バークレー校で開発された論理合成ツール「SIS (Sequential Interactive Synthesis)」を使って実際に論理式から論理回路を生成する過程を体験してもらいます．これにより，論理式の書き方やツールの扱い方しだいで論理合成の結果が大きく変わることを実感していただけるはずです．また，最近になって重要性が増しているタイミング最適化やその他の論理合成の最新事情についても紹介します．これらの内容を，ハードウェア設計の初心者でも理解できるように心がけて，説明していく予定です．

## *1．論理合成の基本技術*

まず，はじめに本稿で使用する記号について説明します．扱うのは基本的に論理変数と論理関数で，‘0’と‘1’の2値をとります．変数名や関数名は，英字1字か，英字1字に数字をつけたもので表現します．たとえば，a，b，c1，f1 (a1，a3) などです．論理積（AND）は基本的に省略し，論理和（OR）は＋と書きます．たとえば，変数aと変数b1のANDは，ab1と書き，ab + c1c2は，変数aと変数bのANDをとったものと変数c1と変数c2のANDをとったもののORをとったものです．また，否定は否定をとるものの後に「'」を付け加えます．たとえば，a'は変数aの否定ですし，(a + b)'は，a + bの否定，つまりa'b'です（ド・モルガンの規則）．

ここでは当面，組み合わせ回路のみを扱います．組み合わせ回路を構成するゲートには，AND，OR，NAND，NOR，NOT，EXOR，EXNORなどの基本ゲートと，複雑な論理関数を直接表現できるAND-OR-INVERTERなどの複合ゲート，さらには，加算器など算術演算に対応したものなどがあります．

論理合成とは，論理式からこのようなゲートで構成される論理回路を自動生成することです．たとえば，**図1**(a)のように，論理式 x = a'b + ab'は，AND，OR，NOTからなるゲートから構成することもできますし，また，排他的論理和（EXOR）ゲートを一つ用いて実現することもできます．また，**図1**(b)のように，x = a + a'bは，直接回路に変換することもできますが，じつはこの論理式は**図1**(c)のようにx = a + bと簡単化でき，結果としてより簡単な回路が得られます．ここで注意していただきたいのは，この二つの回路は等価であるということです．

このように，与えられた論理式（あるいは，与えられた真理値表）をなんらかの方法で「最適化」し，それを回路に変換することになります．前者は論理関数の簡単化と呼ばれ，論理回路がこの世に登場したと同時にさまざまな研究が続けられてきました．本連載ではその代表的な手法を順に，例を用いてわかりやすく説明していきたいと思います．

論理合成の流れを図にすると，**図2**のようになります．まず，ハードウェア記述言語（HDL：hardware description language）で仕様を記述し，それをまず論理式に変換します．その論理式に積和形論理式最適化や多段論理式最